SIAM J. MATRIX ANAL. APPL.
Vol. 23, No. 2, pp. 581–599

STOCHASTIC AUTOMATA NETWORKS AND NEAR COMPLETE DECOMPOSABILITY*

OLEG GUSAK†, TUĞRUL DAYAR†, AND JEAN-MICHEL FOURNEAU‡

Abstract. Stochastic automata networks (SANs) have been developed and used in the last fifteen years as a modeling formalism for large systems that can be decomposed into loosely connected components. In this work, we extend the near complete decomposability concept of Markov chains (MCs) to SANs so that the inherent difficulty associated with solving the underlying MC can be forecasted and solution techniques based on this concept can be investigated. A straightforward approach to finding a nearly completely decomposable (NCD) partitioning of the MC underlying a SAN requires the computation of the nonzero elements of its global generator. This is not feasible for very large systems even in sparse matrix representation due to memory and execution time constraints. We devise an efficient decompositional solution algorithm to this problem that is based on analyzing the NCD structure of each component of a given SAN. Numerical results show that the given algorithm performs much better than the straightforward approach.

Key words. Markov chains, stochastic automata networks, near complete decomposability, state classification

AMS subject classifications. 60J27, 60J10, 65F30, 65F10, 65F50

PII. S089547980036975X

1. Introduction.

Stochastic automata networks (SANs) [16, 18, 13, 17, 19, 21, 22, 9, 1, 6, 12, 24, 3] provide a methodology for modeling large systems with interacting components. The main idea is to decompose the system of interest into its components and to model each component separately. Once this is done, interactions and dependencies among components can be brought into the picture and the model finalized. With this decompositional approach, the global system ends up having as many states as the product of the number of states of the individual components. The benefit of the SAN approach is twofold. First, each component can be modeled much easier compared to the global system due to state space reduction. Second, space required to store the description of components is minimal compared to the case in which transitions from each global state are stored explicitly. However, all this happens at the expense of increased analysis time [13, 22, 1, 9, 6, 12, 24, 3].

An intimately related way of coping with the state space explosion problem is to consider hierarchical decompositions arising in queueing network and superposed stochastic Petri Net formalisms [4, 2, 5]. SANs which do not have dependencies among automata are, in fact, a special case of hierarchical Markovian models. Although somewhat distant from the problem domain compared to the SAN approach, there are recent results showing that hierarchical representations lend themselves naturally to distributed steady state analysis (see [5, p. 79]).

An important issue in choosing an efficient iterative solver for SANs is the conditioning [15] associated with the underlying Markov chain (MC). Recent numerical

*Received by the editors March 27, 2000; accepted for publication (in revised form) by D. O'Leary May 4, 2001; published electronically November 13, 2001. This work was supported by grant TÜBİTAK-CNRS.

http://www.siam.org/journals/simax/23-2/36975.html

†Department of Computer Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey (gusak@cs.bilkent.edu.tr, tugrul@cs.bilkent.edu.tr).

‡Lab. PRiSM, Université de Versailles, 45 Avenue des États-Unis, 78035 Versailles Cedex, France (jmf@prism.uvsq.fr).

experiments [11] show that two-level iterative solvers perform very well with nearly completely decomposable (NCD) partitionings [8] having balanced block sizes when the MC to be solved for its steady state vector is ill-conditioned. Block iterative methods based on classical splittings (block Jacobi, block Gauss–Seidel, block SOR) for SANs are introduced in [24]. Results with iterative aggregation-disaggregation type [23, 20, 10, 11] solvers for SANs appear in [1]. However, two-level iterative solvers considered so far do not exploit NCD partitionings. It should be emphasized that iterative aggregation-disaggregation based on NCD partitionings has certain rate of convergence guarantees [20] that may be useful for very large MCs.

In this paper, we extend the concept of near complete decomposability to SANs so that the inherent difficulty associated with solving the underlying MC can be forecasted and solution techniques based on this concept can be investigated. In doing this, we utilize the graph theoretic ideas for SANs given in [13]. In the next section, we review basic concepts of the SAN formalism and introduce NCD MCs. In section 3, we make assumptions regarding the description of a continuous-time SAN model and discuss how we proceed when we encounter an underlying MC with transient states and/or multiple essential subsets of states. In section 4, we present a three step algorithm that finds an NCD partitioning of the MC underlying a SAN based on a user specified decomposability parameter without computing the global generator matrix. In the first three subsections we discuss the three steps of the proposed algorithm, and in the last subsection we give a summary of its complexity analysis. Numerical results with the algorithm on a SAN model are presented in section 5. We conclude in section 6.

The extended version of this paper can be found in [14]. Therein, we discuss in more detail the approach presented in this paper and provide the algorithms for each of the three steps of the NCD partitioning algorithm introduced here, their detailed complexity analysis, and the results of experiments with two other SAN models.

2. Background. In the next two subsections, we discuss basic concepts related to the SAN formalism as a modeling paradigm and introduce NCD MCs.

2.1. SAN overview. In a SAN (see [21, Chapter 9]), each component of the global system is modeled by a stochastic automaton. When automata do not interact (i.e., when they are independent of each other), description of each automaton consists of local transitions only. In other words, local transitions are those that affect the state of one automaton. Local transitions can be constant (i.e., independent of the state of other automata) or they can be functional. In the latter case, the transition is a nonnegative real valued function that depends on the state of other automata. Interactions among components are captured by synchronizing transitions. Synchronization among automata happens when a state change in one automaton causes a state change in other automata. Similar to local transitions, synchronizing transitions can be constant or functional.

A continuous-time system of N components can be modeled by a single stochastic automaton for each component. Local transitions of automaton $i \in \{1, 2, \ldots, N\}$ (denoted by $\mathcal{A}^{(i)}$) are modeled by the local transition rate matrix $Q_l^{(i)}$. When there are E synchronizing events in the system, $\mathcal{A}^{(i)}$ has the corresponding synchronizing transition matrix $Q_{e_j}^{(i)}$ that represents its contribution to synchronization $j \in \{1, 2, \ldots, E\}$ and associated with it the diagonal corrector matrix $\bar{Q}_{e_j}^{(i)}$. The automaton that triggers a synchronizing event is called the master; the others that get affected by the event are called the slaves. Matrices associated with synchronizing events are either

transition rate matrices (corresponding to master automata) or transition probability matrices (corresponding to slave automata). If $\mathcal{A}^{(i)}$, $i \in \{1, 2, \ldots, N\}$, is not involved in event j, then $Q_{e_j}^{(i)} = \bar{Q}_{e_j}^{(i)} = I_{n_i}$, where n_i is the number of states in $\mathcal{A}^{(i)}$ and I_{n_i} is the identity matrix of order n_i.

The continuous-time Markov chain (CTMC) underlying the global system can be obtained from

$$Q = \bigoplus_{i=1}^{N} Q_l^{(i)} + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)} + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} \bar{Q}_{e_j}^{(i)}, \tag{1}$$

where $\bigoplus$ is the tensor sum operator and $\bigotimes$ is the tensor product operator (see [7]). We refer to the tensor representation in (1) associated with the CTMC as the descriptor of the SAN. When there are functional elements, tensor products become generalized tensor products [19]. Assuming that the states of automata and the global states are numbered starting from 1, the global state s that corresponds to the state vector $(s_1, s_2, \ldots, s_N)$ is given by $s = 1 + \sum_{i=1}^{N}(s_i - 1)\prod_{k=i+1}^{N} n_k$, where $s_i \in \{1, 2, \ldots, n_i\}$ denotes the state of $\mathcal{A}^{(i)}$.

2.2. NCD MCs. NCD MCs [15] are irreducible stochastic matrices that can be symmetrically permuted [8] to the block form

$$P_{n\times n} = \begin{pmatrix} P_{11} & P_{12} & \ldots & P_{1K} \\ P_{21} & P_{22} & \ldots & P_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ P_{K1} & P_{K2} & \ldots & P_{KK} \end{pmatrix}$$

in which the nonzero elements of the off-diagonal blocks are small compared with those of the diagonal blocks [21, p. 286]. Hence, it is possible to represent an NCD MC as

$$P = \mathrm{diag}(P_{11}, P_{22}, \ldots, P_{KK}) + E,$$

where the diagonal blocks P_{ii} are square and possibly of different order. The quantity $\|E\|_\infty$ is referred to as the degree of coupling and is taken to be a measure of the decomposability of P. When the chain is NCD, it has eigenvalues close to 1, and the poor separation of the unit eigenvalue implies a slow rate of convergence for standard matrix iterative methods [10, p. 290]. Hence, NCD MCs are said to be ill-conditioned [15, p. 258]. We should remark that the definition of NCDness is given through a discrete-time Markov chain (DTMC). The underlying CTMC of a SAN can be transformed through uniformization [21, p. 24] to a DTMC for the purpose of computing its steady state vector as in

$$P = I + \frac{1}{\alpha}Q, \tag{2}$$

where $\alpha \geq \max_{1\leq i\leq n} |Q(i,i)|$. To preserve NCDness in this transformation, α must be chosen as $\max_{1\leq i\leq n} |Q(i,i)|$.

An NCD partitioning of P corresponding to a user specified decomposability parameter ϵ can be computed as follows (see [8] for details). First, construct an undirected graph whose vertices are the states of P by introducing an edge between vertices i and j if $P(i,j) \geq \epsilon$ or $P(j,i) \geq \epsilon$, and then identify its connected components[1] (CCs). Each CC forms a subset of the NCD partitioning. Notice that for

[1]Not to be confused with the word *component*, which we have been using so far to mean "subsystem."

a given value of ϵ, the maximum number of subsets in a computed partitioning is unique.

3. On continuous-time SAN descriptions and state classification. There is no standard specification for the description of a SAN model. In the next subsection, we state definitions and propositions that enable us to transform a continuous-time SAN description to one that is more convenient to work with when developing the NCD partitioning algorithm.

3.1. Description of a continuous-time SAN model. Without loss of generality, we restrict ourselves to the case in which row sums of synchronizing transition probability matrices are either 0 or 1.

DEFINITION 1. *A SAN description is said to be proper if and only if each synchronizing transition probability matrix has row sums of* 0 *or* 1.

The SAN descriptions of the three applications we consider in the numerical experiments are proper. However, in a given SAN description, row sums between 0 and 1 can very well be present in synchronizing transition rate matrices. Proposition 1 shows what should be done when such a case is encountered.

PROPOSITION 1. *A given SAN description can be transformed to a SAN description that is proper.*

Proof. Without loss of generality, consider a SAN description of N automata and one synchronizing event. There are two possible cases. In the first case, row sums of the synchronizing transition probability matrix $Q_{e_1}^{(k)}$ corresponding to slave automaton k are all equal to some constant β such that $0 < \beta < 1$. This is the trivial case; we can replace $Q_{e_1}^{(k)}$ with $\hat{Q}_{e_1}^{(k)} = \frac{1}{\beta} Q_{e_1}^{(k)}$ and $Q_{e_1}^{(m)}$ with $\hat{Q}_{e_1}^{(m)} = \beta Q_{e_1}^{(m)}$, where m is the index of the master automaton of the synchronizing event. Row sums of the transformed matrix $\hat{Q}_{e_1}^{(k)}$ are 1. In the second case, row sums of $Q_{e_1}^{(k)}$ are not equal, and some are between 0 and 1. This implies that transition rates of the master automaton m of the synchronizing event depend on the state of automaton k. Therefore, it is possible to replace $Q_{e_1}^{(k)}$ with a matrix that has row sums of 0 or 1 by introducing functional transitions to $Q_{e_1}^{(m)}$ as follows. Let β_l, $l = 1, 2, \ldots, n_k$, be the sum of row l in $Q_{e_1}^{(k)}$. We replace $Q_{e_1}^{(k)}$ with $\hat{Q}_{e_1}^{(k)}$ in which $\hat{Q}_{e_1}^{(k)}(i,j) = Q_{e_1}^{(k)}(i,j)/\beta_i$ if $0 < \beta_i < 1$, else $\hat{Q}_{e_1}^{(k)}(i,j) = Q_{e_1}^{(k)}(i,j)$, for $j = 1, 2, \ldots, n_k$. We also replace $Q_{e_1}^{(m)}$ with $\hat{Q}_{e_1}^{(m)}$ in which $\hat{Q}_{e_1}^{(m)}(i,j) = \beta_l Q_{e_1}^{(m)}(i,j)$ if $0 < \beta_l < 1$, else $\hat{Q}_{e_1}^{(m)}(i,j) = Q_{e_1}^{(m)}(i,j)$, for $i, j = 1, 2, \ldots, n_m$ when $\mathcal{A}^{(k)}$ is in state l. The transformed matrix $\hat{Q}_{e_1}^{(k)}$ has row sums of 0 or 1.

Given a synchronizing event, the above modifications must be made for each of its synchronizing transition probability matrices that has row sums between 0 and 1. After modifying the synchronizing event matrices, the corresponding diagonal corrector matrices must also be modified accordingly. The new SAN description has synchronizing transition probability matrices with row sums of 0 or 1, and therefore is proper.

The generalization to E (> 1) synchronizing events is straightforward. ∎

Observe that the transformation of a SAN description discussed in the proof of Proposition 1 may cause the number of functional elements in the synchronizing transition rate matrices of automata to increase. However, the number of synchronizing events as well as the nonzero structure of the synchronizing transition matrices of automata remain unchanged.

Now we introduce a definition related to the separability of synchronizing transition rates from local transition rates.

DEFINITION 2. *Synchronizations are separable from local transitions in a given SAN description if and only if for any synchronizing event t whose master is automaton m and $i, j = 1, 2, \ldots, n_m$, $Q_{e_t}^{(m)}(i,j) \neq 0$ implies $Q_l^{(m)}(i,j) = 0$.*

Definition 2 may seem to be specifying an artificial condition at first, yet the condition is satisfied by the three applications we consider. As we shall see in the next section, this property enables the preprocessing of local transition rate matrices separately from synchronizing transition matrices which significantly improves the complexity of the NCD partitioning algorithm we propose. Even though the three SAN descriptions we consider have separable synchronizations, one may very well encounter those that do not satisfy this property. Proposition 2 shows that a SAN description whose synchronizations are not separable can be handled in the framework discussed in this paper.

PROPOSITION 2. *A given SAN description can be transformed to a new SAN description whose synchronizations are separable from local transitions.*

Proof. Assume that the given SAN description does not satisfy the condition in Definition 2. Without loss of generality, let t be the event, m its master, and (i,j) the indices of the problematic element. Decompose $Q_l^{(m)}$ into three terms as

$$Q_l^{(m)} = R_l^{(m)} + Q_l^{(m)}(i,j)u_i u_j^T - Q_l^{(m)}(i,j)u_i u_i^T,$$

where u_i is the ith column of the identity matrix. Here $R_l^{(m)}$ is a transition rate matrix; the second term is a matrix with a single nonzero transition rate at element (i,j); and the third term is the diagonal corrector of the second term. Now, let $R_l^{(m)}$ be the local transition rate matrix of automaton m, and introduce the new synchronizing event v with master automaton m; $Q_{e_v}^{(m)} (= Q_l^{(m)}(i,j)u_i u_j^T)$ is the rate matrix associated with automaton m and synchronizing event v, and $\bar{Q}_{e_v}^{(m)}$ $(= -Q_l^{(m)}(i,j)u_i u_i^T)$ is its diagonal corrector. All other matrices corresponding to synchronizing event v are equal to identity. Now, recall the following identity from tensor algebra:

$$\begin{aligned} &A \bigoplus (Q_l^{(m)} + Q_{e_v}^{(m)} + \bar{Q}_{e_v}^{(m)}) \bigoplus B \\ &\quad = \left(A \bigoplus Q_l^{(m)} \bigoplus B\right) + \left(I \bigotimes Q_{e_v}^{(m)} \bigotimes I\right) + \left(I \bigotimes \bar{Q}_{e_v}^{(m)} \bigotimes I\right). \end{aligned}$$

Compare its right-hand side with (1). The new SAN description has separable synchronizations.

The generalization to the cases when event t has more than one problematic element and the SAN description has more than one synchronizing event that are not separable from local transitions is straightforward. ∎

The number of synchronizing events in the new SAN description obtained through the transformation discussed in the proof of Proposition 2 is larger than the number of synchronizing events in the original SAN. The difference in the number of synchronizing events corresponds to the number of the synchronizing events in the original SAN that are not separable. Nevertheless, assuming that identity matrices are not stored explicitly, the described transformation does not increase the number of nonzeros in the transformed SAN description.

Our next definition related to the SAN description involves the number of nonzero elements in synchronizing transition rate matrices. Without loss of generality, we restrict ourselves to the case where all synchronizing events in a SAN are simple.

DEFINITION 3. *Synchronizations in a given SAN description are simple if and only if for any synchronizing event t whose master is automaton m, $Q_{e_t}^{(m)}$ has only one nonzero element.*

In a SAN description whose synchronizations are simple, each synchronizing event can be characterized by a value that corresponds to the synchronizing transition rate of the event. In the next section, we show how to take advantage of this property. In most of the cases, we will not encounter SAN descriptions whose synchronizations are simple. The next proposition shows how SAN descriptions that do not satisfy the condition of Definition 3 can be handled in the framework of our approach.

PROPOSITION 3. *A given SAN description can be transformed to a new SAN description whose synchronizing events are all simple.*

Proof. Assume that the given SAN description does not satisfy the condition in Definition 3. Without loss of generality, let t be the event, m its master, and nz the number of nonzeros in $Q_{e_t}^{(m)}$. Decompose $Q_{e_t}^{(m)}$ into nz simple synchronizing transition rate matrices thereby creating nz new synchronizing events with master automaton m. The slave automata of the new synchronizing events are the slave automata of synchronizing event t. The transition probability matrices and their diagonal correctors associated with the new slave automata are, respectively, equal to the transition probability matrix and its diagonal corrector associated with the slave automata for synchronizing event t. All other matrices corresponding to the new synchronizing events are equal to identity. The new SAN description has simple synchronizations.

The generalization to E (> 1) synchronizing events that are not simple is straightforward. □

Application of the transformation described in the proof of Proposition 3 to a SAN description whose synchronizing events are not simple leads to an increase in the number of synchronizing events. The number of the simple synchronizing events in the new SAN description is equal to the number of nonzero elements in the synchronizing transition rate matrices of the original SAN. Note that the described transformation does not change the synchronizing transition probability matrices and their diagonal correctors. Hence, it is possible to keep the number of nonzero elements in the new SAN description the same as in the original SAN description.

In the next subsection, we discuss how we proceed when we encounter an underlying MC with transient states and/or multiple essential subsets of states.

3.2. State classification in SANs. As discussed in subsection 2.2, NCD MCs are irreducible by definition. However, the MC underlying a SAN may very well be reducible. When the MC underlying the given SAN has transient states and/or multiple essential subsets of states, NCD analysis can be carried out on the essential subsets of states one subset at a time. We name the states that do not belong to the essential subset of interest as uninteresting. We remark that uninteresting states should be omitted from further consideration when running the NCD partitioning algorithm.

We have implemented a state classification (SC) algorithm that classifies the states in the global state space of a SAN into essential and transient subsets following [21, pp. 25–26]. The detailed description of the SC algorithm is given in [14]. The input parameters of the SC algorithm are local transition rate matrices and synchronizing event matrices of the SAN. The output of the algorithm is an integer array of length n in which states corresponding to the essential subset of interest are marked.

4. NCD partitioning algorithm for SANs. The following is our proposed solution algorithm that computes NCD partitionings of the MC underlying a SAN without generating Q (or P).

ALGORITHM 1. NCD PARTITIONING OF MC UNDERLYING SAN FOR GIVEN ϵ.

Step 1. $Q \to P$ transformation.

Step 2. Preprocessing synchronizing events.

Step 3. Constructing NCD connected components.

Step 1 computes the scalar α in (2) that describes the transformation of the global generator Q to a DTMC P through uniformization. In the next subsection, we show how this can be achieved efficiently by inspecting the diagonal elements in local transition rate matrices and the nonzero elements in diagonal corrector matrices.

Step 2 considers the locations of off-diagonal nonzero elements in the global generator Q. Off-diagonal nonzero elements in local transition rate matrices cannot contribute to the same nonzero element in Q due to the fact that these matrices form a tensor sum. Hence, their analysis is straightforward. However, off-diagonal nonzero elements in synchronizing transition rate matrices may contribute to the same nonzero element in Q since these matrices form a sum of tensor products. Therefore, it is necessary to identify those synchronizing events that may influence the NCD partitioning of the MC underlying the SAN by contributing to the value of the same nonzero element in Q. In subsection 4.2, we explain how this is done.

Finally, Step 3 determines the NCD CCs by analyzing local transition rate matrices and matrices corresponding to synchronizing events identified in Step 2 using ϵ and the value of α computed in Step 1. This is discussed in subsection 4.3.

4.1. $Q \to P$ transformation. The CTMC Q can be transformed to a DTMC P using (2) after $\alpha = \max_{1 \le i \le n} |Q(i,i)|$ is computed. Since Q is a CTMC, we have $Q(i,i) = -\sum_{j \neq i} Q(i,j)$ for $i = 1, 2, \ldots, n$. Note also that only the off-diagonal elements in P contribute to NCDness. Regarding the off-diagonal elements in Q, which determine the off-diagonal elements in P, we make the following observations.

REMARK 1. *Each nonzero local transition rate in a SAN contributes to a different off-diagonal element in Q; two or more nonzero local transition rates cannot contribute to the same off-diagonal element in Q.*

This observation follows immediately from the term $\bigoplus_{i=1}^{N} Q_l^{(i)}$ in (1) and the definition of tensor sum.

REMARK 2. *A nonzero off-diagonal element in Q for a SAN with separable synchronizations is formed either of a nonzero local transition rate or of nonzero synchronizing transition rates but not of both.*

This observation follows from the definition of the SAN descriptor in (1) and Definition 2.

From Remarks 1 and 2 and from (1) and (2), P without its main diagonal follows as $P^* = \bigoplus_{i=1}^{N} (\frac{1}{\alpha} Q_l^{(i)}) + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} \hat{Q}_{e_j}^{(i)}$, where $\hat{Q}_{e_j}^{(i)} = \frac{1}{\alpha} Q_{e_j}^{(i)}$ if $\mathcal{A}^{(i)}$ is the master of event j; otherwise, $\hat{Q}_{e_j}^{(i)} = Q_{e_j}^{(i)}$.

REMARK 3. *Dependencies among automata may arise either as explicit functions whose values depend on the states of automata other than the ones in which they are defined or implicitly by the existence of zero rows in synchronizing event matrices associated with slave automata. The latter case corresponds to the disabling of the synchronized transition when the slave automaton is in local state corresponding to the zero row.*

From now on, by dependencies we refer to both explicit and implicit dependencies

as discussed in Remark 3. A naive solution for a SAN having dependencies is to compute explicitly each diagonal element of Q and to find the element with maximum magnitude. However, this is expensive. To reduce the complexity, we propose to partition automata into dependency sets.

DEFINITION 4. *Let $G(\mathcal{V},\mathcal{E})$ be a digraph in which v_i corresponds to $\mathcal{A}^{(i)}$ and $(v_i,v_j)\in\mathcal{E}$ if transitions in $\mathcal{A}^{(i)}$ depend on the state of $\mathcal{A}^{(j)}$ either explicitly or implicitly as discussed in Remark* 3. *Then, the dependency sets of a SAN, denoted by $\mathcal{D}_k$, $k=1,2,\ldots,N_{\mathcal{D}}$, are the connected components of the dependency graph G.*

Assuming that the dependency sets of the SAN are known and referring to

$$\max\_\mathcal{D}_k = \max\left|\bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k}\operatorname{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}}\ \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k}\operatorname{diag}(\bar{Q}_{e_j}^{(i)})\right| \tag{3}$$

as the maximum of the dependency set $\mathcal{D}_k$, where diag returns a vector consisting of the diagonal elements of its matrix argument and $\mathcal{M}_{\mathcal{D}_k}$ is the set of synchronizing events whose masters are in $\mathcal{D}_k$, the diagonal element with maximum magnitude of the MC underlying a SAN can be obtained from

$$\alpha = \sum_{k=1}^{N_{\mathcal{D}}}\max\_\mathcal{D}_k. \tag{4}$$

The proof of this result is given in [14].

Observe that (4) is valid for irreducible MCs underlying SANs. When transient states and/or multiple essential subsets of states are present, the diagonal element with maximum magnitude given by (4) may not belong to the essential subset of interest (see subsection 3.2). In the presence of uninteresting states, we can compute α by finding the maximums of all $N_{\mathcal{D}}$ dependency sets (see (3) and (4)). For dependency set $\mathcal{D}_k$, this task amounts to the enumeration of $\prod_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k} n_i$ states and an equal number of floating-point comparisons. Now, observe that to $\max\_\mathcal{D}_k$ of the dependency set $\mathcal{D}_k$ corresponds a state S_k. Hence, if the global state s that corresponds to $S_1,S_2,\ldots,S_{N_{\mathcal{D}}}$ maps into the essential subset of interest, then α given by (4) is taken as the diagonal element with maximum magnitude. However, if s is an uninteresting state, we omit from further consideration the element corresponding to $\max\_\mathcal{D}_k$ for $k=1,2,\ldots,N_{\mathcal{D}}$ and proceed as in the following paragraph.

In the first step, for $k=1,2,\ldots,N_{\mathcal{D}}$ we find the next largest value denoted by $\text{next\_max\_}\mathcal{D}_k$ from (3) and the corresponding state $\tilde{S}_k$. In order to find $\text{next\_max\_}\mathcal{D}_k$ rapidly, the vectors

$$\left|\bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k}\operatorname{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}}\ \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_k}\operatorname{diag}(\bar{Q}_{e_j}^{(i)})\right|,\quad k=1,2,\ldots,N_{\mathcal{D}},$$

should be stored as sorted. In the second step, we find t such that $\text{next\_max\_}\mathcal{D}_t \geq \text{next\_max\_}\mathcal{D}_k$ for $k=1,2,\ldots,N_{\mathcal{D}}$. Finally, we replace $\max\_\mathcal{D}_t$ with $\text{next\_max\_}\mathcal{D}_t$, S_t with $\tilde{S}_t$, and omit the element corresponding to $\text{next\_max\_}\mathcal{D}_t$ from further consideration. If the updated global state s maps to a state in the essential subset of interest, then α given by (4) is taken as the diagonal element with maximum magnitude. Else we go back to the first step. Since finite MCs have at least one recurrent state in each essential subset, the algorithm is terminating.

Our final remark is about the special case of a SAN with a single dependency set; that is, $N_{\mathcal{D}} = 1$ and $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \ldots, \mathcal{A}^{(N)}\}$. In this case, finding $\alpha = \max\_\mathcal{D}_1$ amounts to enumerating all diagonal elements of Q since we have the equality $\bigoplus_{i,\mathcal{A}^{(i)}\in\mathcal{D}_1} \operatorname{diag}(Q_l^{(i)}) + \sum_{j,e_j\in\mathcal{M}_{\mathcal{D}_k}} \bigotimes_{i,\mathcal{A}^{(i)}\in\mathcal{D}_1} \operatorname{diag}(\bar{Q}_{e_j}^{(i)}) = \operatorname{diag}(Q)$. Therefore, for a SAN with a single dependency set, there is no need to sort and store $\operatorname{diag}(Q)$ as suggested. When finding the maximum of $\operatorname{diag}(Q)$, we test an element of $\operatorname{diag}(Q)$ only if its index corresponds to a state in the essential subset of interest.

Example. This example shows the computation of the diagonal element with maximum magnitude of Q for the following SAN that has functional and synchronizing transitions. The parameters are $N = 3$, $E = 2$, $n_1 = 2$, $n_2 = 3$, $n_3 = 2$; $f = 3$ when $\mathcal{A}^{(1)}$ is in state 1, and $f = 5$ when $\mathcal{A}^{(1)}$ is in state 2. The master of synchronizing event 1 is $\mathcal{A}^{(3)}$, and the master of synchronizing event 2 is $\mathcal{A}^{(2)}$. The matrices are

$$Q_l^{(1)} = \begin{pmatrix} -2 & 2 \\ 1 & -1 \end{pmatrix}, \quad Q_l^{(2)} = \begin{pmatrix} -2 & 2 & 0 \\ 2 & -5 & 3 \\ 1 & 3 & -4 \end{pmatrix}, \quad Q_l^{(3)} = \begin{pmatrix} -f & f \\ 0 & 0 \end{pmatrix},$$

$$Q_{e_1}^{(1)} = \bar{Q}_{e_1}^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \quad Q_{e_2}^{(1)} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \bar{Q}_{e_2}^{(1)} = I,$$

$$Q_{e_1}^{(2)} = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \bar{Q}_{e_1}^{(2)} = I, \quad Q_{e_2}^{(2)} = \begin{pmatrix} 0 & 0 & 5 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \bar{Q}_{e_2}^{(2)} = \begin{pmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

$$Q_{e_1}^{(3)} = \begin{pmatrix} 0 & 0 \\ 5 & 0 \end{pmatrix}, \quad \bar{Q}_{e_1}^{(3)} = \begin{pmatrix} 0 & 0 \\ 0 & -5 \end{pmatrix}, \quad Q_{e_2}^{(3)} = \bar{Q}_{e_2}^{(3)} = I.$$

The given SAN has two dependency sets: $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(3)}\}$ and $\mathcal{D}_2 = \{\mathcal{A}^{(2)}\}$. Note that $\mathcal{A}^{(3)}$ functionally depends on the state of $\mathcal{A}^{(1)}$ due to functional transition f as well as due to synchronizing event 1 (see $\bar{Q}_{e_1}^{(1)}$). Hence, the diagonal element with maximum magnitude of Q is comprised of two terms. The maximum of $\mathcal{D}_1$ is given by

$$\begin{aligned} \max\_\mathcal{D}_1 &= \max\left|\operatorname{diag}(Q_l^{(1)}) \bigoplus \operatorname{diag}(Q_l^{(3)}) + \operatorname{diag}(\bar{Q}_{e_1}^{(1)}) \bigotimes \operatorname{diag}(\bar{Q}_{e_1}^{(3)})\right| \\ &= \max\left|\begin{pmatrix} -2-f \\ -2-0 \\ -1-f \\ -1-3 \end{pmatrix} + \begin{pmatrix} 0 \\ -5 \\ 0 \\ 0 \end{pmatrix}\right| = \max\left|\begin{pmatrix} -5 \\ -2 \\ -6 \\ -4 \end{pmatrix} + \begin{pmatrix} 0 \\ -5 \\ 0 \\ 0 \end{pmatrix}\right| = 7. \end{aligned}$$

On the other hand, $\mathcal{D}_2$ is a singleton, and therefore the maximum of $\mathcal{D}_2$ is given by

$$\max\_\mathcal{D}_2 = \max\left|\operatorname{diag}(Q_l^{(2)}) + \operatorname{diag}(\bar{Q}_{e_2}^{(2)})\right| = \max\left|\begin{pmatrix} -2 \\ -5 \\ -4 \end{pmatrix} + \begin{pmatrix} -5 \\ 0 \\ 0 \end{pmatrix}\right| = 7.$$

Since the underlying MC is irreducible, $\alpha = \max\_\mathcal{D}_1 + \max\_\mathcal{D}_2 = 14$ as verified on

$$Q = \begin{pmatrix}
-12 & 3 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 5 & 0 \\
0 & -14 & 0 & 2 & 5 & 0 & 0 & 2 & 0 & 0 & 0 & 5 \\
2 & 0 & -10 & 3 & 3 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\
5 & 2 & 0 & -12 & 0 & 3 & 0 & 0 & 0 & 2 & 0 & 0 \\
1 & 0 & 3 & 0 & -9 & 3 & 0 & 0 & 0 & 0 & 2 & 0 \\
5 & 1 & 0 & 3 & 0 & -11 & 0 & 0 & 0 & 0 & 0 & 2 \\
1 & 0 & 0 & 0 & 5 & 0 & -13 & 5 & 2 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 5 & 0 & -8 & 0 & 2 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & -11 & 5 & 3 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & 0 & -6 & 0 & 3 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 3 & 0 & -10 & 5 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 3 & 0 & -5
\end{pmatrix}.$$

As pointed out at the beginning of this subsection, an NCD partitioning of P that corresponds to a user specified decomposability parameter ϵ is determined by the off-diagonal elements in P. In the next subsection we concentrate on those off-diagonal elements that originate from the synchronizing transition rates of the SAN.

4.2. Preprocessing synchronizing events. Transition rates from different synchronizing event matrices may sum up to form a nonzero in the generator matrix Q. Hence, in some cases it may not be possible to determine the value of an off-diagonal element in Q by inspecting each automaton separately. The aim of Step 2 in Algorithm 1 is to find sets of those synchronizing events that may influence the NCD partitioning of Q. We name these sets as potential sets of synchronizing events. The potential sets are disjoint, and their union is a subset of the set of synchronizing events. The input parameters of Step 2 are synchronizing event matrices, ϵ, and α computed in Step 1. The output of Step 2 is $N_{\mathcal{P}}$ potential sets denoted by $\mathcal{P}_r$, $r = 1, 2, \ldots, N_{\mathcal{P}}$.

There are two cases in which synchronizing events may influence the NCD partitioning of Q. First, a simple synchronizing event has the corresponding transition rate greater than or equal to $\alpha\epsilon$. Second, a set of synchronizing events contribute to the same element in Q, and the sum of the synchronizing transition rates of the events in the set is greater than or equal to $\alpha\epsilon$.

In the first case, each synchronizing event with transition rate greater than or equal to $\alpha\epsilon$ forms a potential set that is a singleton. When the transition rate of a synchronizing event is a function, its value can be evaluated only on the global state space. This can be done in Step 3 of Algorithm 1 when NCD CCs of the SAN are formed. Hence, if the synchronizing transition rate is a function and the maximum value of the function is not known in advance, then the corresponding synchronizing event also forms a potential set that is a singleton. Regarding the second case, we make the following observation. The position of a synchronizing transition rate in Q is uniquely determined by all synchronizing transition matrices that correspond to the synchronizing event. This can be seen from (1). Hence, we have the following proposition.

PROPOSITION 4. *In a SAN with simple synchronizations, the set $\mathcal{E}^*$ of synchronizing events contribute to the same nonzero element of Q if and only if there exists at least one nonzero element with the same indices in the matrices $Q_{e_j}^{(i)}$ for all $e_j \in \mathcal{E}^*$ and $i = 1, 2, \ldots, N$.*

Proof. The proof follows from (1), the definition of tensor product, and Definitions 2 and 3. □

Those synchronizing events that are not classified as potential sets of singletons must be tested for the condition in Proposition 4. The test of two events, t and u, for the condition requires the comparison of the indices of nonzero elements in $Q_{e_t}^{(i)}$ and $Q_{e_u}^{(i)}$ for $i = 1, 2, \ldots, N$; that is, we test N pairs of matrices. For k events, the number of matrix pairs that need to be tested is $Nk(k-1)/2$. Note that for three events, t, u, and v, the fact that the pairs $(Q_{e_t}^{(i)}, Q_{e_u}^{(i)})$ and $(Q_{e_u}^{(i)}, Q_{e_v}^{(i)})$ each have at least one nonzero element with the same indices for $i = 1, 2, \ldots, N$ does not imply that the events t and v also satisfy the condition. In other words, the condition is not transitive. This further complicates the test for the condition in Proposition 4.

In order to avoid excessive computation associated with the test, we consider the set of synchronizing events $\mathcal{P}$ as a potential set if for all $e_u \in \mathcal{P}$ there exists $e_v \in \mathcal{P}$ such that the condition in Proposition 4 is satisfied for synchronizing events u and v, and the sum of transition rates of synchronizing events in $\mathcal{P}$ is greater than or equal to $\alpha\epsilon$. According to this definition, we form potential sets as follows. Let $\mathcal{L}$ be the set of synchronizing events that are not classified as potential sets of singletons. We choose event $e_v \in \mathcal{L}$, remove it from $\mathcal{L}$, and test e_v with each event in $\mathcal{L}$ for the condition in Proposition 4. Let $\mathcal{K}$ be the set of events that satisfy this condition. Then, if the sum of the transition rates of synchronizing event v and those in $\mathcal{K}$ is greater than or equal to $\alpha\epsilon$, we remove the events that are in $\mathcal{K}$ from $\mathcal{L}$ and form the potential set $\mathcal{P} = \{e_v\} \cup \mathcal{K}$. We repeat this procedure for all events in $\mathcal{L}$ until $\mathcal{L} = \emptyset$.

Example (*continued*). Let $\epsilon = 0.3$, implying $\alpha\epsilon = 4.2$. The transition rate of the master automaton of simple synchronizing event 1 is 5 and greater than $\alpha\epsilon$ (see $Q_{e_1}^{(3)}(2,1)$). Hence, the first potential set, $\mathcal{P}_1$, consists of synchronizing event 1 only. The second synchronizing event of the SAN also forms a potential set. See $Q_{e_2}^{(2)}(1,3)$ for justification. Thus, $\mathcal{P}_1 = \{e_1\}$ and $\mathcal{P}_2 = \{e_2\}$. Now, consider the case in which $\epsilon = 0.4$, implying $\alpha\epsilon = 5.6$. Both transition rates of synchronizing events 1 and 2 are less than $\alpha\epsilon$. Hence, we have to test these two events for the condition in Proposition 4; that is, we check if each of the three pairs of matrices $(Q_{e_1}^{(1)}, Q_{e_2}^{(1)})$, $(Q_{e_1}^{(2)}, Q_{e_2}^{(2)})$, and $(Q_{e_1}^{(3)}, Q_{e_2}^{(3)})$ have at least one nonzero element with the same indices. However, the condition in Proposition 4 is not satisfied. Thus, the number of potential sets for the case of $\epsilon = 0.4$ is zero. This implies that neither of the synchronizing events influence the NCD partitioning of the underlying MC. Therefore, when $\epsilon = 0.4$, synchronizing events of the SAN are omitted from further consideration in Step 3 of Algorithm 1.

4.3. Constructing NCD connected components. As indicated in Remark 2, a nonzero element in the global generator of a SAN originates either from a local transition rate or from one or more synchronizing transition rates. Hence, NCD CCs of the underlying MC are determined by (i) constant local transition rates that are greater than or equal to $\alpha\epsilon$, (ii) functional local transition rates that can take values greater than or equal to $\alpha\epsilon$, or (iii) transition rates of synchronizing events that are in the potential sets $\mathcal{P}_r$, $r = 1, 2, \ldots, N_{\mathcal{P}}$. These three different possibilities are considered in Step 3 of Algorithm 1. The input parameters of Step 3 are local transition rate matrices and synchronizing event matrices, ϵ, α computed in Step 1, and potential sets formed in Step 2. The output of Step 3 is the set of NCD CCs of the underlying MC.

First, we consider possibility (i) in which local transition rates are constant, and assume that $Q = Q_l$ (see (1)). Using $\alpha\epsilon$, we can find the NCD CCs of $Q_l^{(i)}$, $i =$

$1, 2, \ldots, N$. Let $\mathcal{C}^{(i)}$ be the set of NCD CCs of $Q_l^{(i)}$, where a member of $\mathcal{C}^{(i)}$, denoted by $\mathtt{c}^{(i)}$, is a partition of states from $\mathcal{A}^{(i)}$. Let $\mathcal{B}$ and $\mathcal{H}$ be sets in which each member of either set is also a set. In other words, $\mathcal{B}$ as well as $\mathcal{H}$ is a set of sets. We define the binary operator $\odot$ between the two sets $\mathcal{B}$ and $\mathcal{H}$ as $\mathcal{B} \odot \mathcal{H} = \{\mathtt{b} \times \mathtt{h} \mid \mathtt{b} \in \mathcal{B}, \mathtt{h} \in \mathcal{H}\}$, where $\times$ is the ordinary Cartesian product operator. Then, based on the graph interpretation of the tensor sum operator discussed in [13], the set of NCD CCs is given by $\mathcal{C} = \mathcal{C}^{(1)} \odot \mathcal{C}^{(2)} \odot \cdots \odot \mathcal{C}^{(N)}$. Observe that if $\mathcal{C}^{(i)}$, $i = 1, 2, \ldots, N$, are singletons, then $\mathcal{C}$ is a singleton as well; that is, the underlying MC is not NCD for given ϵ. One can take advantage of the same property when there are only K ($< N$) $\mathcal{C}^{(i)}$ that are singletons. In this case, we renumber the automata so that these K sets assume indices from $(N - K + 1)$ to N. Then these K sets can be replaced with the set $\mathcal{C}^{[N-K+1]} = \{\{1, 2, \ldots, n_{N-K+1}\} \times \{1, 2, \ldots, n_{N-K}\} \times \cdots \times \{1, 2, \ldots, n_N\}\}$.

Now we bring into the picture functional local transition rates and consider possibility (ii). Let us assume that the automata of the given SAN can be reordered and renumbered so that transitions of automaton i depend (if at all) on the states of higher indexed automata, but they do not depend on the states of lower indexed automata (see [12] for details). Since Cartesian product is associative, $\odot$ is also associative, and one can rewrite the expression for $\mathcal{C}$ as

$$\mathcal{C} = \left(\mathcal{C}^{(1)} \odot \left(\mathcal{C}^{(2)} \odot \cdots \odot \left(\mathcal{C}^{(N-1)} \odot \mathcal{C}^{(N)}\right) \cdots\right)\right). \tag{5}$$

Given $\mathcal{C}^{[k]} = (\mathcal{C}^{(k)} \odot (\mathcal{C}^{(k+1)} \odot \cdots \odot (\mathcal{C}^{(N-1)} \odot \mathcal{C}^{(N)}) \cdots))$, the union of all members of $\mathcal{C}^{[k]}$ is a set that is equivalent to the product state space of $\mathcal{A}^{(k)}, \mathcal{A}^{(k+1)}, \ldots, \mathcal{A}^{(N)}$. Therefore, taking into account the assumed ordering of automata, functional transition rates of $\mathcal{A}^{(k)}$ can be evaluated and NCD CCs of $\mathcal{C}^{[k]}$ can be updated accordingly. More formally, let $Q_l^{(k)}(s_k, \tilde{s}_k)$ be a functional element, i.e., $Q_l^{(k)}(s_k, \tilde{s}_k) = f$. Then the NCD CCs $\mathtt{c}^{[k]}, \tilde{\mathtt{c}}^{[k]} \in \mathcal{C}^{[k]}$ must be joined if $(s_k, s_{k+1}, \ldots, s_N) \in \mathtt{c}^{[k]}$, $(\tilde{s}_k, s_{k+1}, \ldots, s_N) \in \tilde{\mathtt{c}}^{[k]}$, and $f(s_k, s_{k+1}, \ldots, s_N) \geq \alpha\epsilon$.

Example (*continued*). We illustrate possibilities (i) and (ii) on the SAN description by omitting synchronizing events 1 and 2. Synchronizing events are treated in possibility (iii). We set $\epsilon = 0.3$ implying $\alpha\epsilon = 4.2$ and assume that the automata are ordered as $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. First, we find the NCD CCs of all local transition rate matrices as in possibility (i) by treating functional transition rates as zero. Inspection of local transition rate matrices shows that local transition rates of all automata are less than $\alpha\epsilon$. Hence, we have $\mathcal{C}^{(1)} = \{\{1_1\}, \{2_1\}\}$, $\mathcal{C}^{(2)} = \{\{1_2\}, \{2_2\}, \{3_2\}\}$, and $\mathcal{C}^{(3)} = \{\{1_3\}, \{2_3\}\}$. The subscripts in the states enable us to distinguish between states with identical indices but that belong to different automata. According to (5), we form the NCD CCs of $Q_l^{(3)} \bigoplus Q_l^{(1)}$, i.e., $\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)} = \{\{(1_3, 1_1)\}, \{(1_3, 2_1)\}, \{(2_3, 1_1)\}, \{(2_3, 2_1)\}\}$. Then we continue with possibility (ii). The value of the functional transition rate $Q_l^{(3)}(1, 2)$ ($= f$) depends on the state of $\mathcal{A}^{(1)}$ only. Hence, we can evaluate f when $\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)}$ is formed. The functional transition rate f evaluates to 5, which is larger than $\alpha\epsilon$, when $\mathcal{A}^{(1)}$ is in state 2. Therefore, we join $\{(1_3, 2_1)\}$ and $\{(2_3, 2_1)\}$. Finally, the NCD CCs of Q_l are given by

$$\begin{aligned}\mathcal{C} &= \mathcal{C}^{(2)} \odot (\mathcal{C}^{(3)} \odot \mathcal{C}^{(1)}) = \{\{1_2\}, \{2_2\}, \{3_2\}\} \odot \{\{(1_3, 1_1)\}, \{(1_3, 2_1), (2_3, 2_1)\}, \{(2_3, 1_1)\}\} \\ &= \{\{(1_2, 1_3, 1_1)\}, \{(1_2, 2_3, 1_1)\}, \{(2_2, 1_3, 1_1)\}, \{(2_2, 2_3, 1_1)\}, \{(3_2, 1_3, 1_1)\}, \{(3_2, 2_3, 1_1)\}, \\ &\quad \{(1_2, 1_3, 2_1), (1_2, 2_3, 2_1)\}, \{(2_2, 1_3, 2_1), (2_2, 2_3, 2_1)\}, \{(3_2, 1_3, 2_1), (3_2, 2_3, 2_1)\}\}.\end{aligned}$$

Now we consider possibility (iii). When possibilities (i) and (ii) are handled, the union of all members in $\mathcal{C}$ is a set that corresponds to the global state space of

the SAN. The transition rate of synchronizing event t can be taken into account as follows. Let $(s_1, s_2, \ldots, s_N) \in \mathtt{c}$ and $(\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_N) \in \tilde{\mathtt{c}}$, where $\mathtt{c}, \tilde{\mathtt{c}} \in \mathcal{C}$. Then $\mathtt{c}$ and $\tilde{\mathtt{c}}$ must be joined if $\prod_{i=1}^{N} Q_{e_t}^{(i)}(s_i, \tilde{s}_i) \geq \alpha\epsilon$. Since the global state space of the SAN is usually very large, it may take a significant amount of time to find all pairs $\mathtt{c}$ and $\tilde{\mathtt{c}}$ that satisfy this condition. Fortunately, the situation can be improved. Let p, $1 < p \leq N$, be the smallest index among automata involved in event t, i.e., $Q_{e_t}^{(i)} = I_{n_i}$ for $i = 1, 2, \ldots, p-1$. We rewrite the first two terms of (1) as

$$\bigoplus_{i=1}^{N} Q_l^{(i)} + \sum_{j=1}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)} = \left(\bigoplus_{i=1}^{p-1} Q_l^{(i)} \right) \bigoplus Q_l^{[p]} + \left(\bigotimes_{i=1}^{p-1} I_{n_i} \right) \bigotimes Q_{e_t}^{[p]} + \sum_{j=1, j \neq t}^{E} \bigotimes_{i=1}^{N} Q_{e_j}^{(i)}, \tag{6}$$

where $Q_l^{[p]} = \bigoplus_{i=p}^{N} Q_l^{(i)}$ and $Q_{e_t}^{[p]} = \bigotimes_{i=p}^{N} Q_{e_t}^{(i)}$. From the definition of tensor sum, the first two terms of expression (6) can be written as

$$\left(\bigoplus_{i=1}^{p-1} Q_l^{(i)} \right) \bigoplus Q_l^{[p]} + \left(\bigotimes_{i=1}^{p-1} I_{n_i} \right) \bigotimes Q_{e_t}^{[p]} = \left(\bigoplus_{i=1}^{p-1} Q_l^{(i)} \right) \bigoplus \left(Q_l^{[p]} + Q_{e_t}^{[p]} \right). \tag{7}$$

From (7), it can be seen that the transition rate of synchronizing event t can be taken into account on the smaller state space $\mathcal{C}^{(p)} \odot \mathcal{C}^{(p+1)} \odot \cdots \odot \mathcal{C}^{(N)}$. The same idea can be extended to the potential sets formed in Step 2. In other words, if for $\mathcal{P}_r$, there exists σ_r, $1 < \sigma_r \leq N$, such that $Q_{e_j}^{(i)} = I_{n_i}$ for $i = 1, 2, \ldots, \sigma_r - 1$ and all $e_j \in \mathcal{P}_r$, then transition rates of synchronizing events in $\mathcal{P}_r$ can be taken into account when the set $\mathcal{C}^{[\sigma_r]} = \mathcal{C}^{(\sigma_r)} \odot \mathcal{C}^{(\sigma_r+1)} \odot \cdots \odot \mathcal{C}^{(N)}$ is formed. We remark that for the assumed ordering of automata, all functional transitions that may be present in synchronizing transition matrices of events in $\mathcal{P}_r$ can be evaluated when $\mathcal{C}^{[\sigma_r]}$ is formed.

Example (*continued*). For $\epsilon = 0.3$, each of the two synchronizing events of the SAN is classified as a potential set. We assume the same ordering of automata, i.e., $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. After renumbering the automata, let the new indices of the automata be $\tilde{1}$, $\tilde{2}$, $\tilde{3}$, respectively. For the given ordering of automata, the smallest index among automata involved in event 1 as well as in event 2 is $\tilde{1}$. Hence, the transition rates of events 1 and 2 can be taken into account when $\mathcal{C}^{[\tilde{1}]} = \mathcal{C}$ is formed. Due to the transition rate of synchronizing event 1, we join the NCD CCs that have the members $(1_2, 2_3, 1_1)$ and $(3_2, 1_3, 1_1)$, $(2_2, 2_3, 1_1)$ and $(1_2, 1_3, 1_1)$, $(3_2, 2_3, 1_1)$ and $(1_2, 1_3, 1_1)$. Similarly, due to synchronizing event 2, we join the NCD CCs that have the members $(1_2, 1_3, 1_1)$ and $(3_2, 1_3, 2_1)$, $(1_2, 1_3, 2_1)$ and $(3_2, 1_3, 1_1)$, $(1_2, 2_3, 1_1)$ and $(3_2, 2_3, 2_1)$, $(1_2, 2_3, 2_1)$ and $(3_2, 2_3, 1_1)$. For justification, see $\mathcal{C}$ formed in the example following possibility (ii) and the SAN description.

When the automata of a SAN have cyclic dependencies, they cannot be ordered as discussed. Such cases can be handled as follows. Let $G(\mathcal{V}, \mathcal{E})$ be the digraph in which v_i corresponds to $\mathcal{A}^{(i)}$ and $(v_i, v_j) \in \mathcal{E}$ if transitions in $\mathcal{A}^{(i)}$ depend on the state of $\mathcal{A}^{(j)}$ (see Definition 4). Let G_{SCC} be the digraph obtained by collapsing each SCC of G to a single vertex. This graph is acyclic and the automata of the SAN can be ordered topologically with respect to G_{SCC}. Assuming that the automata are in this order, let p be the smallest index among cyclically dependent automata. Then we can evaluate all functions in the cyclically dependent automata when $\mathcal{C}^{[p]}$ is formed. The special case in which a cyclic dependency is created by transitions in the synchronizing transition matrices of a particular event can be handled in the same way as discussed in possibility (iii). There, the potential set $\mathcal{P}_r$, $r \in \{1, 2, \ldots, N_{\mathcal{P}}\}$, is taken into account when $\mathcal{C}^{[\sigma_r]}$ is formed. Assuming that the automata are ordered

topologically with respect to G_{SCC}, all functions in the matrices of synchronizing events that belong to $\mathcal{P}_r$ can be evaluated when $\mathcal{C}^{[\sigma_r]}$ is formed.

Our final remark is about a SAN with more that one essential subset of states and/or transient states. For $1 < i \leq N$, we do not have a one-to-one mapping between the global state space and the union of all members in $\mathcal{C}^{[i]}$. Hence, we cannot say whether a member of $\mathtt{c}^{[i]} \in \mathcal{C}^{[i]}$ maps to a state in the essential subset of interest or to an uninteresting state. Therefore, the decomposition of $\mathcal{C}$ as in (5) that allows us to handle functional local transition rates and synchronizing transition rates on a smaller state space cannot be used. This is because one or both of the members that belong to the joined NCD CCs may map to an uninteresting state. For a SAN with uninteresting states, possibilities (ii) and (iii) should be considered on the global state space. Hence, the NCD CCs $\mathtt{c}, \tilde{\mathtt{c}} \in \mathcal{C}$ should be joined only if the members under consideration from each of the two sets map into the essential subset of interest. When we compute $\mathcal{C} = \mathcal{C}^{(1)} \odot \mathcal{C}^{(2)} \odot \cdots \odot \mathcal{C}^{(N)}$, uninteresting states must also be omitted from consideration. From the definition of the binary operator $\odot$, if s_i and $\tilde{s}_i$ are in the same NCD CC of $\mathcal{C}^{(i)}$, then it must be that $(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1} \ldots, s_N)$ and $(s_1, s_2, \ldots, s_{i-1}, \tilde{s}_i, s_{i+1} \ldots, s_N)$ are in the same NCD CC of $\mathcal{C}$. When uninteresting states are present, we exercise the additional constraint that $(s_1, s_2, \ldots, s_{i-1}, s_i, s_{i+1} \ldots, s_N)$ and $(s_1, s_2, \ldots, s_{i-1}, \tilde{s}_i, s_{i+1} \ldots, s_N)$ must belong to the essential subset of interest.

In the next subsection, we summarize for Algorithm 1 the detailed space and time complexity analysis that appears in [14] and apply the results to our example.

4.4. Complexity analysis of Algorithm 1. The core operation performed by an algorithm that finds the NCD CCs of a MC is floating-point comparison. Hence, we provide the number of floating-point comparisons performed in Algorithm 1. Regarding the algorithm's storage requirements, we remark that its three steps are executed sequentially. Hence, the maximum amount of memory required by Algorithm 1 is upper bounded by an integer array of length $O(n)$.

For the sake of simplicity, we assume that the MC underlying the SAN is irreducible. In Step 1, the number of floating-point comparisons is given by $\sum_{k=1}^{N_D} \prod_{i, \mathcal{A}^{(i)} \in \mathcal{D}_k} n_i$. For the best case in which each dependency set is a singleton, the number of floating-point comparisons reduces to $\sum_{i=1}^{N} n_i$. On the other hand, if all automata form a single dependency set, we have the upper bound $\prod_{i=1}^{N} n_i = n$. In Step 2, the lower bound on the number of floating-point comparisons is E, and it corresponds to the case in which the transition rate of each simple synchronizing event is greater than or equal to $\alpha\epsilon$. The upper bound is equal to $\frac{1}{2}E(E+1)$ floating-point comparisons. This number of floating-point comparisons is achieved when the transition rate of each simple synchronizing event is less than $\alpha\epsilon$ and the transition rates of synchronizing events do not sum up in Q. The number of floating-point comparisons in Step 3 depends strongly on the number of functional transitions and synchronizing events as well as the automata ordering. Assuming that in Step 2 of Algorithm 1 synchronizing event r is classified as the potential set $\mathcal{P}_r$, $r = 1, 2, \ldots, E$, and the automata are ordered as discussed in possibility (ii) in subsection 4.3, the number of floating-point comparisons in Step 3 is given by

$$\sum_{i=1}^{N} nz_l^{(i)} + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j \neq m_r}^{N} nz_{e_r}^{(j)},$$

where $nz_l^{(i)}$ is the number of nonzero off-diagonal elements in $Q_l^{(i)}$, nf_i is the number

of functional transitions in $Q_l^{(i)}$, $nz_{e_r}^{(j)}$ is the number of nonzeros in $Q_{e_r}^{(i)}$, and m_r is the index of the master automaton of event r. Finally, the number of floating-point comparisons performed in Algorithm 1 is given by

$$E + \sum_{i=1}^{N}(n_i + nz_l^{(i)}) + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$$

in the best case, and

$$n + \frac{1}{2}E(E+1) + \sum_{i=1}^{N} nz_l^{(i)} + \sum_{i=1}^{N-1} nf_i \prod_{j=i+1}^{N} n_j + \sum_{r=1}^{E} \prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$$

in the worst case.

Step 3 of Algorithm 1 also incurs floating-point multiplications when synchronizing events are handled. Computation of a single nonzero transition originating from synchronizing event r requires $(N - \sigma_r)$ floating-point multiplications. For synchronizing event r, we compute $\prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}$ elements. Hence, the maximum number of floating-point multiplications in Step 3 is $\sum_{r=1}^{E}[(N - \sigma_r)\prod_{j=\sigma_r, j\neq m_r}^{N} nz_{e_r}^{(j)}]$. Observe that this expression is almost the same as the last term of the expression for the number of floating-point comparisons performed in Algorithm 1. Hence, assuming that the time it takes to perform floating-point multiplication and floating-point comparison are of the same order, the time complexity of Algorithm 1 is roughly the number of floating-point comparisons.

Example (*continued*). We calculate the number of floating-point comparisons performed by Algorithm 1 to find an NCD partitioning of the MC underlying the SAN. We use the same input parameters for Algorithm 1 as in subsection 4.3; that is, $\epsilon = 0.3$ and the automata are ordered as $\mathcal{A}^{(2)}$, $\mathcal{A}^{(3)}$, $\mathcal{A}^{(1)}$. Following the three steps of Algorithm 1 on our example, we see that Step 1 takes $n_1 n_3 + n_2 = 7$ floating-point comparisons to find the maximums of 2 dependency sets, and Step 2 takes 2 floating-point comparisons to form the 2 potential sets of singletons. Step 3 takes 7+2+3+4=16 floating-point comparisons, where 7 is the number of comparisons to find $\mathcal{C}^{(1)}$, $\mathcal{C}^{(2)}$, $\mathcal{C}^{(3)}$; 2 is the number of comparisons to handle the functional local transition of $\mathcal{A}^{(3)}$; and 3 and 4 are the numbers of comparisons to process transition rates of synchronizing events 1 and 2, respectively. Thus, the total number of floating comparisons performed in Algorithm 1 is 25. The number of floating-point multiplications performed to process synchronizing events 1 and 2 is $(N-1)(nz_{e_1}^{(1)} nz_{e_1}^{(2)} + nz_{e_2}^{(1)} nz_{e_2}^{(3)}) = 14$. When the global generator is stored in sparse format, the total number of floating-point comparisons performed by the straightforward algorithm that finds NCD CCs of Q is 57, which is almost two times as large as the corresponding value of Algorithm 1.

5. Numerical results. We implemented the SC algorithm and Algorithm 1 in C++ as part of the software package PEPS [18]. We ran all the experiments on a Sun UltraSparcstation 10 with 128 MBytes of RAM. To verify the NCD partitionings obtained for a given SAN, we compared our results with the straightforward approach of generating in core the submatrix of Q corresponding to the essential subset of states obtained using the SC algorithm and finding its NCD CCs. We remark that the same data structure for NCD CCs is used in Algorithm 1 and the straightforward approach.

The input parameters of Algorithm 1 are the user specified decomposability parameter ϵ, the vector output by the SC algorithm in which states corresponding to the

essential subset of interest are marked, and a file in PEPS format that contains the description of the SAN under consideration and the dependencies among automata. We remark that the only modification that we make on the SAN description is the transformation of each synchronizing event to the simple form (if the SAN is not already in that form). Note that this transformation is taken into account in the reported results.

As test problems, we use the three SAN models that appear in [24]. We name them resource sharing, three queues, and mass storage. Here, we present the results of experiments with the three queues problem. The results of experiments with the other two problems appear in [14]. The SAN model of the three queues problem consists of four automata $\mathcal{A}^{(1)}, \mathcal{A}^{(2)}, \mathcal{A}^{(3_1)}, \mathcal{A}^{(3_2)}$ with, respectively, C_1, C_2, C_3, C_3 states and two synchronizing events. The state space size is given by $n = C_1 C_2 C_3^2$ and there is a single subset of $C_1 C_2 C_3 (C_3 + 1)/2$ essential states. Functional transition rates appear in local transition rate and synchronizing event matrices. There are two dependency sets $\mathcal{D}_1 = \{\mathcal{A}^{(1)}, \mathcal{A}^{(3_1)}, \mathcal{A}^{(3_2)}\}$ and $\mathcal{D}_2 = \{\mathcal{A}^{(2)}\}$. Detailed description of the three queues problem and its parameters can be found in [12]. In our experiments, we use the values of real parameters in [24].

Results of experiments for the three queues problem are presented in Table 1. All timing results are in seconds. In Table 1, n denotes the number of states in the global state space of the particular SAN under consideration, n_{ess} denotes the number of states in the essential subset, nz_{ess} denotes the number of nonzero elements in the submatrix of Q corresponding to the essential subset of states, and SC denotes the time for state classification. For each problem, we indicate in parentheses under n the values of the integer parameters used. The column ϵ denotes the value of the decomposability parameter used and $|CCs|$ denotes the number of NCD CCs corresponding to ϵ when transient states are removed. The column NCD_S contains timing results for Algorithm 1. The columns Gen. and NCD_N, respectively, contain timing results to generate in core the submatrix of Q corresponding to the essential subset of states and to naively compute its NCD partitioning for given ϵ after the SC algorithm is executed. We have varied the value of ϵ in each problem to see how the performance of Algorithm 1 changes for different number of NCD CCs.

We remark that the difference between the time required to generate in core the submatrix of Q corresponding to the essential subset of states for a given SAN and the time to find the corresponding NCD partitionings using Algorithm 1 is noticeable. Compare columns Gen. and NCD_S, and also compare the sum of columns Gen. and NCD_N with column NCD_S. Moreover, there are cases for which it is not possible to generate in core the submatrix of Q corresponding to the essential subset of states on the particular architecture. Hence, the straightforward approach of finding NCD partitionings is relatively more restricted with memory and is slower than using Algorithm 1.

The time spent for state classification does not involve any floating-point operations, whereas the time spent to generate in core the submatrix of Q corresponding to the essential subset of states primarily involves floating-point arithmetic operations. The overhead associated with evaluating functions slows down both tasks dramatically. Compare columns SC and Gen. with columns NCD_S and NCD_N. The time spent by the SC algorithm is larger than the time spent by Algorithm 1 in all experiments. This is not surprising since the former is based on finding SCCs while the latter is based on finding CCs. The difference is more pronounced when there are multiple dependency sets for which Algorithm 1 can bring in considerable savings.

TABLE 1
Results of the three queues problem (C_1, C_2, C_3).

n	n_{ess}	nz_{ess}	SC	ϵ	$\lvert CCs\rvert$	NCD_S	Gen.	NCD_N
68,850	36,720	207,279	0.82	0.10	1	0.10	0.39	0.05
(18,17,15)				0.22	544	0.22		0.09
				0.25	4,590	0.13		0.07
				0.35	36,720	0.11		0.06
202,400	106,260	608,474	2.63	0.10	1	0.30	1.24	0.17
(23,22,20)				0.22	924	0.68		0.28
				0.25	10,120	0.38		0.24
				0.35	106,260	0.33		0.23
756,000	390,600	2,264,460	9.83	0.10	1	1.03	4.58	0.62
(30,28,30)				0.22	1,652	2.46		1.04
				0.25	25,200	1.37		0.92
				0.35	390,600	1.12		0.90
1,414,875	727,650	4,239,795	19.04	0.10	1	1.88	8.37	1.16
(35,33,35)				0.22	2,277	4.60		1.94
				0.25	40,425	2.46		1.71
				0.35	727,650	2.02		1.56
6,875,000	3,506,250	20,632,250	96.37	0.10	1	8.63		
(50,55,50)				0.22	5,445	21.85		
				0.25	137,500	11.57		
				0.35	3,506,250	9.18		
9,150,625	4,658,500	27,445,825	131.34	0.10	1	11.25		
(55,55,55)				0.22	5,995	33.04		
				0.25	166,375	14.24		
				0.35	4,658,500	12.44		

The case of $|CCs| = 1$ corresponds to smaller ϵ and implies the largest number of nonzeros taken into account from automata matrices in Algorithm 1 and from the submatrix of Q corresponding to the essential subset of states in the naive NCD partitioning algorithm. The case of $|CCs| = n_{\text{ess}}$ corresponds to larger ϵ and implies larger temporary data structures being used by both algorithms when determining NCD CCs. Hence, for increasing ϵ, the results in columns NCD_S and NCD_N either increase then decrease.

6. Conclusion. In this work, we have considered the application of the near complete decomposability concept to SANs. The definitions, propositions, and remarks presented in sections 3 and 4 have enabled us to devise an efficient algorithm that computes NCD partitionings of the MC underlying a SAN. The approach is based on determining the NCD connected components of a SAN from the description of individual automata without generating the global transition rate matrix. We have also implemented a state classification algorithm for SANs that classifies each state in the global state space as essential or transient. The output of the state classification algorithm is used in the NCD partitioning algorithm for SANs. The time and space complexities of the NCD partitioning algorithm depend on the number of automata, the number of synchronizing events, the number of functions, the number of essential states of interest, the sparsity of automata matrices, the dependency sets, and the ordering of automata. Future work should focus on taking advantage of the partitionings computed by the devised algorithms in two-level iterative solvers.

Acknowledgments. We thank the anonymous referees for their constructive remarks, which led to an improved manuscript.

REFERENCES

[1] P. Buchholz, *An aggregation/disaggregation algorithm for stochastic automata networks*, Probab. Engrg. Inform. Sci., 11 (1997), pp. 229–253.

[2] P. Buchholz, *An adaptive aggregation/disaggregation algorithm for hierarchical Markovian models*, European J. Oper. Res., 116 (1999), pp. 545–564.

[3] P. Buchholz, *Projection methods for the analysis of stochastic automata networks*, in Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains, B. Plateau, W. J. Stewart, and M. Silva, eds., Prensas Universitarias de Zaragoza, Zaragoza, Spain, 1999, pp. 149–168.

[4] P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper, *Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models*, INFORMS J. Comput., 12 (2000), pp. 203–222.

[5] P. Buchholz, M. Fischer, and P. Kemper, *Distributed steady state analysis using Kronecker algebra*, in Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains, B. Plateau, W. J. Stewart, and M. Silva, eds., Prensas Universitarias de Zaragoza, Zaragoza, Spain, 1999, pp. 76–95.

[6] R. H. Chan and W. K. Ching, *Circulant preconditioners for stochastic automata networks*, Numer. Math., 87 (2000), pp. 35–57.

[7] M. Davio, *Kronecker products and shuffle algebra*, IEEE Trans. Comput., C-30 (1981), pp. 116–125.

[8] T. Dayar, *Permuting Markov Chains to Nearly Completely Decomposable Form*, Technical Report BU-CEIS-9808, Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey, 1998; also available online from ftp://ftp.cs.bilkent.edu.tr/pub/tech-reports/1998/BU-CEIS-9808.ps.z.

[9] T. Dayar, O. I. Pentakalos, and A. B. Stephens, *Analytical Modeling of Robotic Tape Libraries Using Stochastic Automata*, Technical Report TR-97-189, CESDIS, NASA/GSFC, Greenbelt, MD, 1997.

[10] T. Dayar and W. J. Stewart, *On the effects of using the Grassmann–Taksar–Heyman method in iterative aggregation–disaggregation*, SIAM J. Sci. Comput., 17 (1996), pp. 287–303.

[11] T. Dayar and W. J. Stewart, *Comparison of partitioning techniques for two-level iterative solvers on large, sparse Markov chains*, SIAM J. Sci. Comput., 21 (2000), pp. 1691–1705.

[12] P. Fernandes, B. Plateau, and W. J. Stewart, *Efficient descriptor-vector multiplications in stochastic automata networks*, J. ACM, 45 (1998), pp. 381–414.

[13] J.-M. Fourneau and F. Quessette, *Graphs and stochastic automata networks*, in Computations with Markov Chains: Proceedings of the 2nd International Workshop on the Numerical Solution of Markov Chains, W. J. Stewart, ed., Kluwer, Boston, MA, 1995, pp. 217–235.

[14] O. Gusak, T. Dayar, and J.-M. Fourneau, *Stochastic Automata Networks and Near Complete Decomposability*, Technical Report BU-CE-0016, Department of Computer Engineering, Bilkent University, Ankara, Turkey, 2000; also available online from ftp://ftp.cs.bilkent.edu.tr/pub/tech-reports/2000/BU-CE-0016.ps.z.

[15] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems*, SIAM Rev., 31 (1989), pp. 240–272.

[16] B. Plateau, *On the stochastic structure of parallelism and synchronization models for distributed algorithms*, in Proceedings of the SIGMETRICS Conference on Measurement and Modelling of Computer Systems, Austin, TX, 1985, pp. 147–154.

[17] B. Plateau and K. Atif, *Stochastic automata network for modeling parallel systems*, IEEE Trans. Software Engrg., 17 (1991), pp. 1093–1108.

[18] B. Plateau, J.-M. Fourneau, and K.-H. Lee, *PEPS: A package for solving complex Markov models of parallel systems*, in Modeling Techniques and Tools for Computer Performance Evaluation, R. Puigjaner and D. Ptier, eds., Palma de Mallorca, Spain, 1988, pp. 291–305.

[19] B. Plateau and J.-M. Fourneau, *A methodology for solving Markov models of parallel systems*, J. Parallel Distrib. Comput., 12 (1991), pp. 370–387.

[20] G. W. Stewart, W. J. Stewart, and D. F. McAllister, *A two-stage iteration for solving nearly completely decomposable Markov chains*, in Recent Advances in Iterative Methods, IMA Vol. Math. Appl. 60, G. H. Golub, A. Greenbaum, and M. Luskin, eds., Springer-Verlag, New York, 1994, pp. 201–216.

[21] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[22] W. J. Stewart, K. Atif, and B. Plateau, *The numerical solution of stochastic automata networks*, European J. Oper. Res., 86 (1995), pp. 503–525.

[23] W. J. Stewart and W. Wu, *Numerical experiments with iteration and aggregation for Markov*

chains, ORSA J. Comput., 4 (1992), pp. 336–350.
[24] E. Uysal and T. Dayar, *Iterative methods based on splittings for stochastic automata networks*, European J. Oper. Res., 110 (1998), pp. 166–186.